CS 106B Section 5 (Week 6) Solutions

1. Big-O

$O(N^3)$

```
int numA(Grid<char> board) {    // Solution: O(N * N * N)
    Vector<int> total;                                      // O(1)
    for (int row = 0; row < board.numRows(); row++) {       // O(N * N * N)
        for (int col = 0; col < board.numCols(); col++) {       // O(N * N)
            for (int rowT = 0; rowT <= row; rowT++) {               // O(N)
                for (char ch = 'a'; ch <= 'z'; ch++) {                  // O(1)
                    if (ch == 'a' && rowT == row && board[row][col] == ch) {    // O(1)
                        total.add(38);                                              // O(1)
                    }
                }
            }
        }
    }
    return total.size();                                    // O(1)
}
```

$O(N^3)$

```
int numARecursive(Grid<char> &board) {  // Called max N times, so O(N * (N * N))
    int total = 0;                                          // O(1)
    for (int i = 0; i < board.numRows(); i++) {             // O(N)
        total += (board[i][0] == 'a');                          // O(1)
    }
    if (board.numCols() == 1) {                             // O(1)
        return total;                                           // O(1)
    }
    Grid<char> newBoard(board.numRows(), board.numCols() - 1);  // O(1)
    for (int i = 0; i < board.numRows(); i++) {             // O(N * N)
        for (int j = 0; j < board.numCols() - 1; j++) {         // O(N)
            newBoard[i][j] = board[i][j + 1];                       // O(1)
        }
    }
    return total + numARecursive(newBoard);                 // O(N)
}
```

CS 106B Section 5 (Week 6) Solutions

2. Sorting

```
void cocktailSort(Vector<int> &vec) {
    bool sorted = false;
    bool forward = true;
    while (!sorted) {
        sorted = true;
        for (int i = 0; i < vec.size() - 1; i++) {
            int firstIndex;
            if (forward) {
                firstIndex = i;
            } else {
                firstIndex = vec.size() - i - 2;
            }
            int secondIndex = firstIndex + 1;
            if (vec[firstIndex] > vec[secondIndex]) {
                int temp = vec[firstIndex];
                vec[firstIndex] = vec[secondIndex];
                vec[secondIndex] = temp;
                sorted = false;
            }
        }
        forward = !forward;
    }
}
```

---

3. Classes

```
void Fraction::reciprocal() {
    int tempDenom = denom;
    denom = num;
    num = tempDenom;

    reduce();
}

void Fraction::divide(Fraction other) {
    mult(Fraction(other.denom, other.num));
}
```

---

4. Pointers

```
1 -84 2
```